TOPIC NO.: 5

TITLE: Implementation of Stack

Program Code:

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10

int c = 0;

struct stack {

int a[SIZE];

int top;

};

typedef struct stack st;

void ces(st \*s) {

s->top = -1;

}

int isoverflow(st \*s) {

if (s->top == SIZE - 1)

return 1;

else

return 0;

}

int isunderflow(st \*s) {

if (s->top == -1)

return 1;

else

return 0;

}

void push(st \*s, int e) {

if (isoverflow(s)) {

printf("Stack Overflow!\n");

}

else {

s->top++;

s->a[s->top] = e;

}

c++;

}

void pop(st \*s) {

if (isunderflow(s)) {

printf("Stack Underflow!\n");

}

else {

printf("popped element is %d\n", s->a[s->top]);

s->top--;

}

c--;

}

DATE: .01.24

void printstack(st \*s) {

printf("Current stack : \n");

for (int i = 0; i < c; i++) {

printf("%d\n", s->a[i]);

printf("\n");

}

}

int main() {

st \*s = (st \*)malloc(sizeof(st));

int choice, element;

while (1) {

printf("Enter 1 to Create a stack\n");

printf("Enter 2 to Push an element\n");

printf("Enter 3 to Pop an element\n");

printf("Enter 4 to Print the stack\n");

printf("Enter 5 to Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

ces(s);

break;

case 2:

printf("Enter the element to push: ");

scanf("%d", &element);

push(s, element);

break;

case 3:

pop(s);

break;

case 4:

printstack(s);

break;

case 5:

free(s);

printf("Exiting the program.\n");

return 0;

default:

printf("Invalid choice!\n");

}

}

return 0;

}

Output:

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 1

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 2

Enter the element to push: 3

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 2

Enter the element to push: 4

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 2

Enter the element to push: 5

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 2

Enter the element to push: 6

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 4

3

4

5

6

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 3

popped element is 6

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack.

5. Exit

Enter your choice: 4

Current stack:

3

4

5

1. Create a stack

2. Push an element

3. Pop an element

4. Print the stack

5. Exit

Enter your choice: 5

Exiting the program.

TOPIC NO.: 6

TITLE: Implementation of Linked List

Program Code:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node \*createNode(int data) {

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Memory Allocation Failed!!\n");

exit(1);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void insertAtBeginning(struct Node \*\*head, int data) {

struct Node \*newNode = createNode(data);

newNode->next = \*head;

\*head = newNode;

}

void insertAtEnd(struct Node \*\*head, int data) {

struct Node \*newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

return;

}

struct Node \*temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

void deleteNode(struct Node \*\*head, int key) {

struct Node \*temp = \*head;

struct Node \*prev = NULL;

if (temp != NULL && temp->data == key) {

\*head = temp->next;

free(temp);

return;

}

while (temp != NULL && temp->data != key) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("Key Not Found In The List\n");

return;

}

prev->next = temp->next;

free(temp);

}

void display(struct Node \*head) {

struct Node \*temp = head;

while (temp!= NULL) {

printf("%d -> ", temp->data);

temp = temp->next;

}

printf("NULL");

}

DATE: .01.24

void reverse(struct Node \*\*head) {

struct Node \*prev = NULL;

struct Node \*current = \*head;

struct Node \*next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

\*head = prev;

}

int main() {

struct Node \*head = NULL;

int choice, data, key;

while (1) {

printf("\n\n1. Insert At Beginning\n");

printf("2. Insert At End\n");

printf("3. Delete A Node\n");

printf("4. Display The List\n");

printf("5. Reverse Yhe List\n");

printf("6. Exit\n");

printf("Enter Your Choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter Data to Insert At Beginning: ");

scanf("%d", &data);

insertAtBeginning(&head, data);

break;

case 2:

printf("Enter Data to Insert At End: ");

scanf("%d", &data);

insertAtEnd(&head, data);

break;

case 3:

printf("Enter Data To Delete: ");

scanf("%d", &key);

deleteNode(&head, key);

break;

case 4:

printf("List: ");

display(head);

break;

case 5:

printf("Reversed List: ");

reverse(&head);

display(head);

reverse(&head);

break;

case 6:

exit(0);

default:

printf("Invalid Choice!! Please enter a valid choice.\n");

}

}

return 0;

}

Output:

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 1

Enter Data to Insert At Beginning: 3

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 1

Enter Data to Insert At Beginning: 5

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 4

List: 5 -> 3 -> NULL

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 3

Enter Data To Delete: 5

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 4

List: 3 -> NULL

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 2

Enter Data to Insert At End: 9

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 2

Enter Data to Insert At End: 4

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 1

Enter Data to Insert At Beginning: 6

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 4

List: 6 -> 3 -> 9 -> 4 -> NULL

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 5

Reversed List: 4 -> 9 -> 3 -> 6 -> NULL

1. Insert At Beginning

2. Insert At End

3. Delete A Node

4. Display The List

5. Reverse Yhe List

6. Exit

Enter Your Choice: 6